Ali ELABRIDI

Supervisor: Pr. Yousra Chtouki

Report of the honor project application related to NoSQL

Introduction/Goals:

The small NoSQL project application that I decided to realize as part of my research component for the honor program as a demonstration of how NoSQL take form and how it can be manipulated is a web application that uses JSON as database in which all the student IDs and their names of Al Akhawayn University are mapped in the form of JSON key/value objects to extract relevant data about them, and to enhance the search feature that is unfortunately not well implemented by the Information Technology Department, and therefore facilitate the lookups by different criterion such as the ID, first name and last name or only part of them.

Procedure:

The first goal was to obtain the data about the university students. To do so, I parsed the data available publically in the Ground Maintenance Website (https://gm.aui.ma/SiteBS/index.php) using regular expressions to look for pattern in the data, and transform the data into a JSON structure. The result of the cleaning data can be found in the following link: https://github.com/alielabridi/AUI-IDs/blob/master/AUI_IDs.json


```
}, {
    "ID": "60303",
    "Firstname": "Hind",
    "Lastname": "Kadiri",
    "Lastname": "amani"
}, {
    "ID": "60304",
    "Firstname": "SOFIA",
    "Lastname": "Benhaddou"
}, {
    "ID": "60310",
    "Firstname": "Anas",
    "Lastname": "Bennani"
}, {
    "ID": "60332",
    "Firstname": "Med Kamal ",
    "Lastname": "Zaraba"
}, {
    "ID": "60341",
    "Firstname": "Rida",
    "Lastname": "El Boustani"
}, {
    "ID": "60352",
    "Firstname": "Naima",
    "Lastname": "Bendahman"
}, {
```


Figure: A visual of how the JSON file is structured

The second objective was to create a user interface in which the end user can interact with to search over this JSON structured data. The web application is written in HTML and JavaScript with the use of the two frameworks: AngularJS and JQuery. HTML is responsible for laying out the basic visual structure of the interface including input field, input button and result table. JavaScript underlies the whole dynamic search part of the program. In which first, I implemented (.contains) method that is much more enhanced than what you can find in the library of the String Object in Javacript. AngularJS is responsible for regenerating the display of the result table dynamically each time the user enters a new search input. JQuery is responsible for getting the JSON file and parsing it into a readable and native JavaScript Object that can be manipulated as any internal variable. The application does not use any "dynamic language" that needs special care such as PHP, Ruby, NodeJS… it uses only HTML and JavaScript. Therefore, the resulting product can be used in any system that has a browser with JavaScript enabled. The code for the web application can be found in the following link:
https://github.com/alielabridi/AUI-IDs/blob/master/index.html

Challenges:

The only challenges encountered in the use of JSON is to disable loading asynchronously the data, and use instead a synchronized one as demonstrated by this piece of code. To make sure that there is no inconsistency in loading the data, and the whole system waits till the json file is downloaded completely before proceeding into manipulating the data.

```
/*force getJSON func
$.ajaxSetup({
async: false
});
```

Conclusion:

JSON as a NoSQL database that takes the form of key/value pairs is a nice and light solution to small applications in which there is no relevant relation between the different attributes or entities. It takes the form a simple file that you can export and use easily in different platform and language as demonstrated with the same JSON file that holds the student ID in an android application of http://scantosign.con that is based in Java instead of JavaScript. Therefore, JSON is a great example of database to be used in small structure If you want to reduce the cost of storing the data since it does not need any specific software or vendor to be purchased from, and the whole structure is open source and free to use.

[BONUS POINTS]

Apply Constraints and policies in a Restful web service:

Motivation:

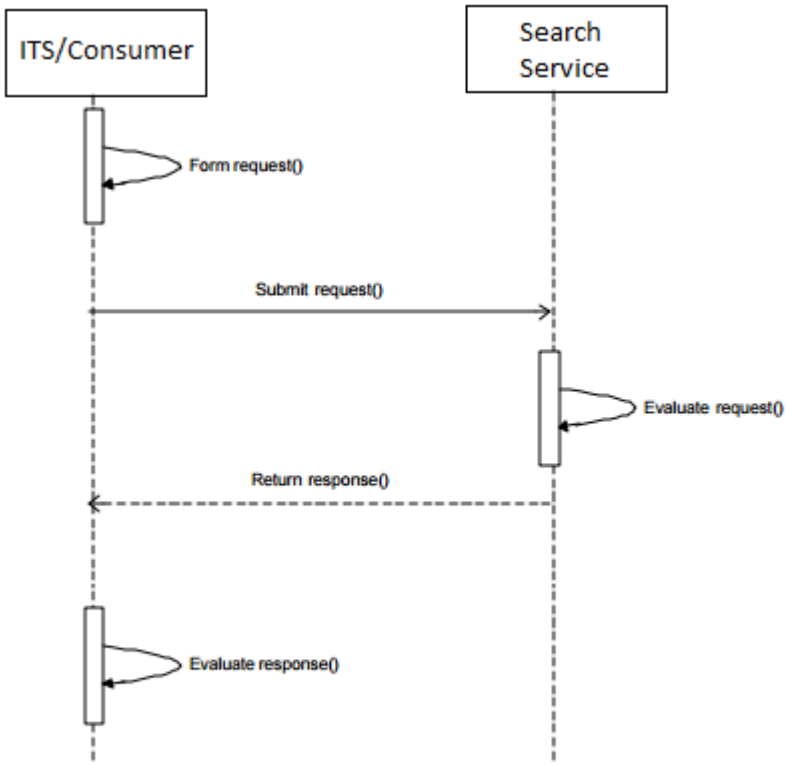
With the use of cloud computing, there is a great demand of externalizing services to other external clients either by providing information (GET) or using information provided for manipulation (POST). However, these interaction between the two or more independent web services need to be bounded by policies and constraints not only to provide a secure and reliable channel between the two, but also to export these policies as generic to other alike web services. The motivation of the experiment is to avoid using WS-policies implemented using a SOAP structure that can be extremely costly and heavy for small uses and quick implementations, but instead use a light Restful API that would mimic the same behavior using a slightly less complex structure but as secure and strong as a WS-Policy.

Goals:

As a reminder, the sole purpose of this experiment is not to produce a rigorous research on WS-policies or how to implement them equivalently using REST since there is no solution out there that implements policies in a rest environment as soap does with WS-policies, but to produce a possible but usable alternative for web services to generate a composed service. We will use as a possible implementation the Student ID searcher introduced earlier and enforce on it constraints on certain uses by an external web services. we will assume that there is a second party external and totally independent from our web service: the ITS department. Every semester there is new students that register at Al Akhawayn University and therefore need to be added to our list of students in our service to be available for search. The possible constraints that can be implemented are the following:

- Source verification
- Authentication & credentials verification
- Integrity of the request according to a predefined structure
- Validity of the data provided i.e. type checking, and range checking

The communication sent back as a response to the request made by the external web service is in the form of HTTP status codes (http://www.restapitutorial.com/httpstatuscodes.html) as it is conventionally used in REST to express the successfulness of the operation or the different errors that may arises when checking again the different constraints e.g. 200 OK status code for success, 401 unauthorized status code for failing to authenticate, 406 Not Acceptable Status Code for checking against the structure and type checking.



Implementation/Conclusion:

The demonstration of the use and the walk-through the code are in the form of a Youtube video that can be found in the link: https://youtu.be/IZNT842RyqE and the Implementation code is free of use and can be found in the Github Repository https://github.com/alielabridi/AUI-IDs/

I hope that you will enjoy the original research and findings and the experiments and its usefulness in a working environment. The work was produced for the Honor component of Database System at Al Akhawayn University Supervised by Pr. Yousra Chtouki in the 7 july 2016

References :

https://www.oasis-open.org/committees/download.php/1608/wd-xacml-wspl-use-cases-04.pdf

http://www.restapitutorial.com/httpstatuscodes.html

http://www.wseas.org/multimedia/journals/education/2015/a105710-150.pdf

http://ijcsi.org/papers/IJCSI-10-5-1-208-218.pdf